LAB 6 REPORT

Course: CPS633

Section: 6

Group Members:

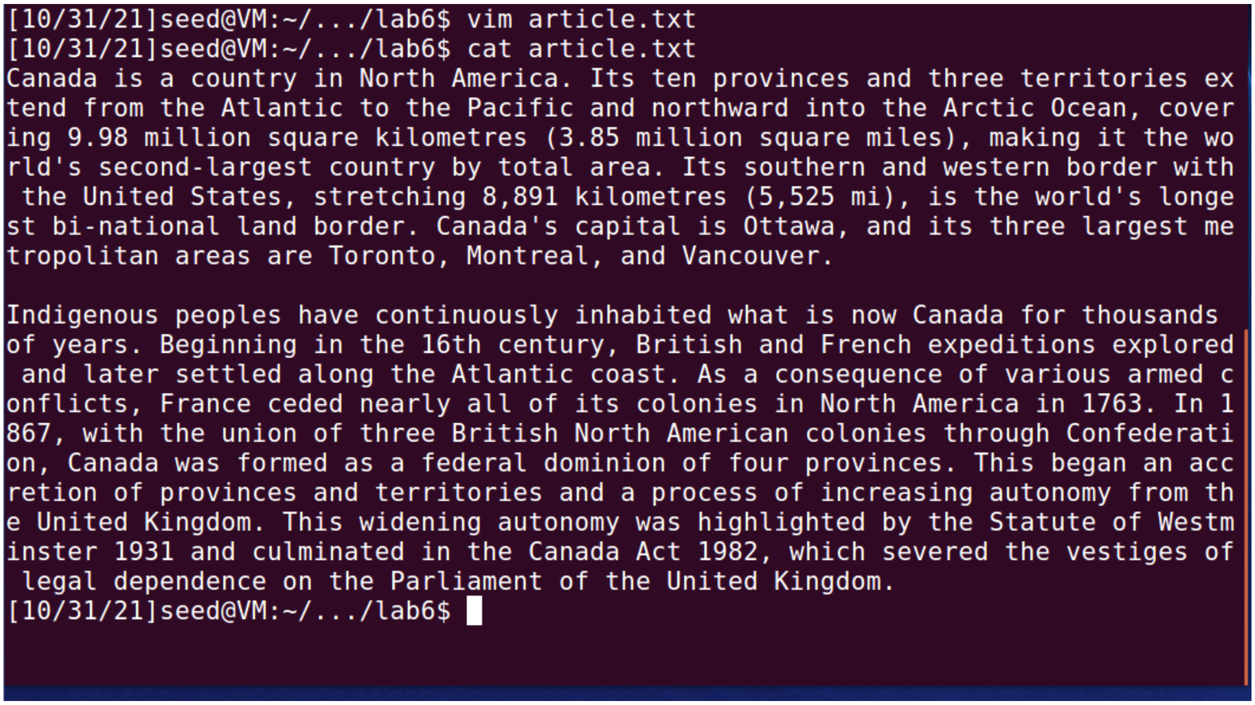
Alishba Aamir, 500974648

Ivan Golovine, 500813431

Fangbo Ren, 500884730

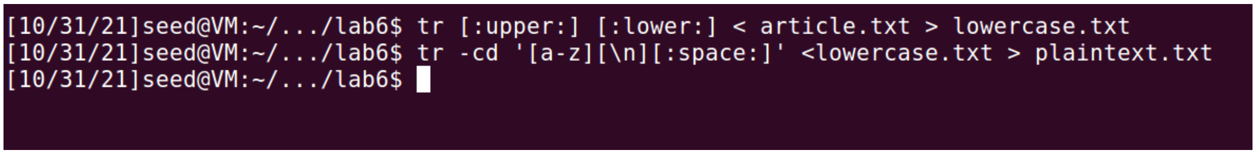
Task 1: Frequency Analysis

First, we make our article that is going to be encrypted.



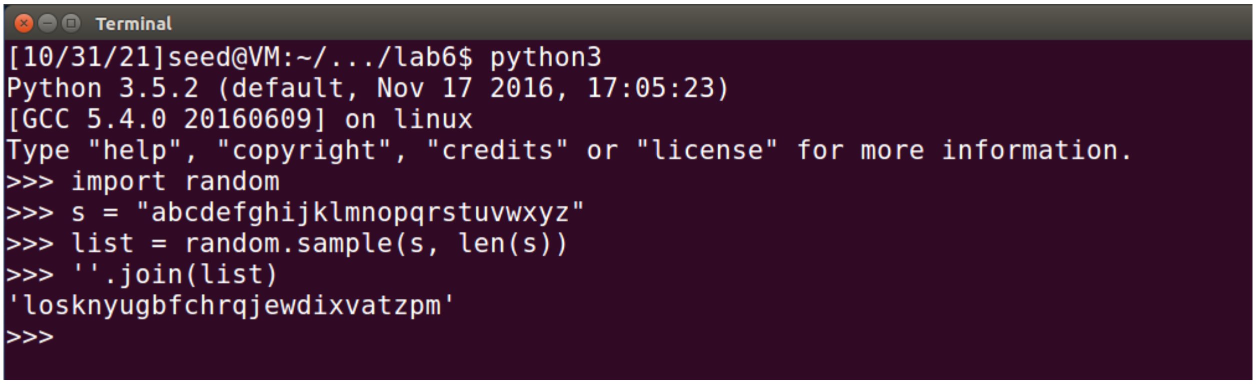
```
[10/31/21]seed@VM:~/.../lab6$ vim article.txt
[10/31/21]seed@VM:~/.../lab6$ cat article.txt
Canada is a country in North America. Its ten provinces and three territories ex
tend from the Atlantic to the Pacific and northward into the Arctic Ocean, cover
ing 9.98 million square kilometres (3.85 million square miles), making it the wo
rld's second-largest country by total area. Its southern and western border with
 the United States, stretching 8,891 kilometres (5,525 mi), is the world's longe
st bi-national land border. Canada's capital is Ottawa, and its three largest me
tropolitan areas are Toronto, Montreal, and Vancouver.

Indigenous peoples have continuously inhabited what is now Canada for thousands
of years. Beginning in the 16th century, British and French expeditions explored
 and later settled along the Atlantic coast. As a consequence of various armed c
onflicts, France ceded nearly all of its colonies in North America in 1763. In 1
867, with the union of three British North American colonies through Confederati
on, Canada was formed as a federal dominion of four provinces. This began an acc
retion of provinces and territories and a process of increasing autonomy from th
e United Kingdom. This widening autonomy was highlighted by the Statute of Westm
inster 1931 and culminated in the Canada Act 1982, which severed the vestiges of
 legal dependence on the Parliament of the United Kingdom.
[10/31/21]seed@VM:~/.../lab6$ 
```

For simplification, we change all the upper case letter in the article to lower case and we keep the spaces.



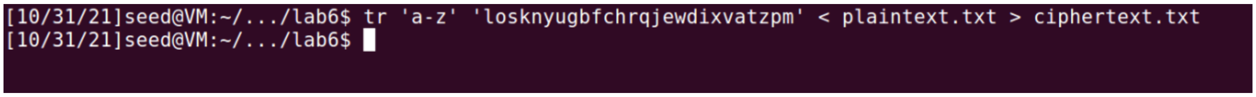
```
[10/31/21]seed@VM:~/.../lab6$ tr [:upper:] [:lower:] < article.txt > lowercase.txt
[10/31/21]seed@VM:~/.../lab6$ tr -cd '[a-z][\n][:space:]' <lowercase.txt > plaintext.txt
[10/31/21]seed@VM:~/.../lab6$ 
```

We use the python program given in the lab manual to generate a substitution key. This program permutes each letter from a to z to a random letter.



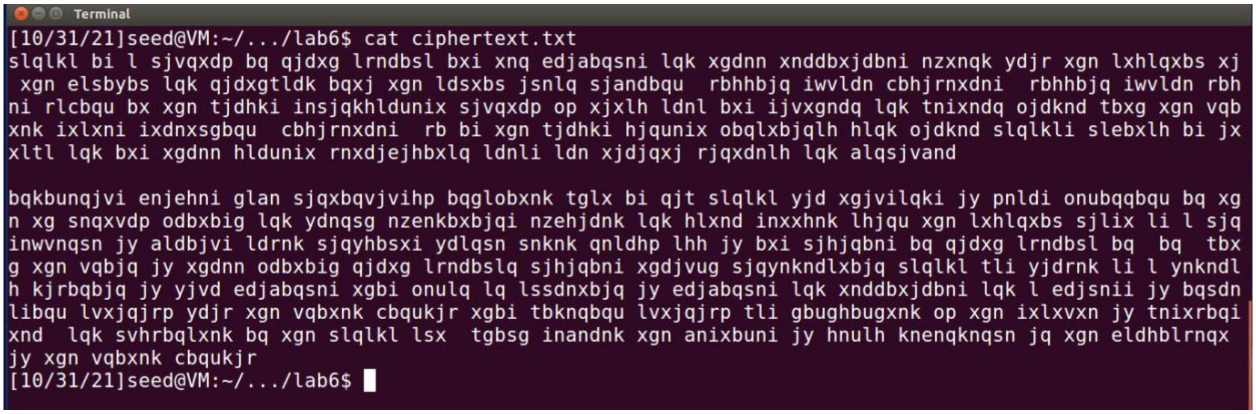
```
[10/31/21]seed@VM:~/.../lab6$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> s = "abcdefghijklmnopqrstuvwxyz"
>>> list = random.sample(s, len(s))
>>> ''.join(list)
'losknyugbfchrqjewdixvatzpm'
>>> 
```

We use this key to encrypt our plain text to cipher text



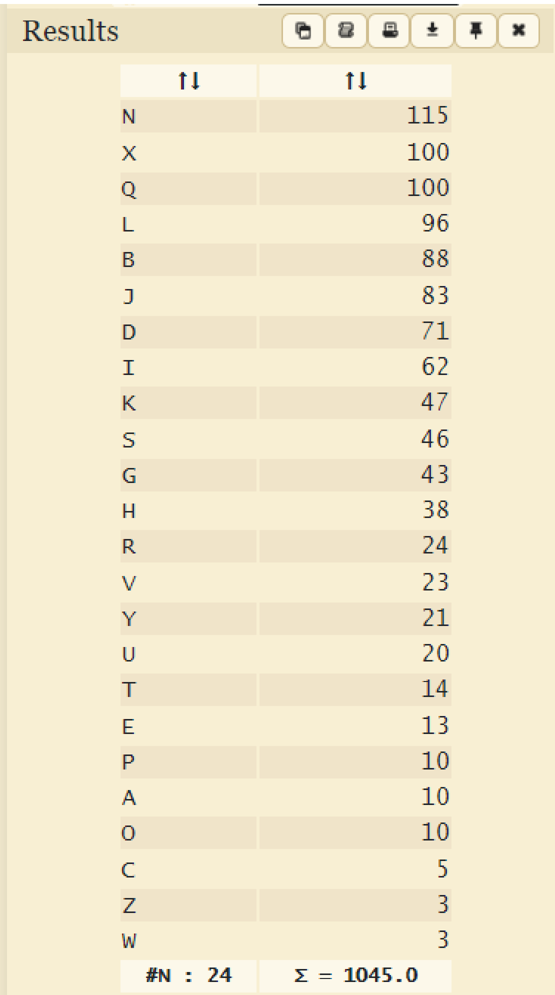
```
[10/31/21]seed@VM:~/.../lab6$ tr 'a-z' 'losknyugbfchrqjewdixvatzpm' < plaintext.txt > ciphertext.txt
[10/31/21]seed@VM:~/.../lab6$ █
```

Below, is our encrypted text.



```
[10/31/21]seed@VM:~/.../lab6$ cat ciphertext.txt
slqlkl bi l sjvqxdp bq qjdxg lrndbsl bxi xnq edjabqsni lqk xgdnn xnddbxjdbni nzxnqk ydjr xgn lxhlqxbs xj
 xgn elsbybs lqk qjdxgtldk bqxj xgn ldsxbs jsnlq sjandbqu  rbhhbjq iwvldn cbhjrnxdni  rbhhbjq iwvldn rbh
ni rlcbqu bx xgn tjdhki insjqkhldunix sjvqxdp op xjxlh ldnl bxi ijvxgndq lqk tnixndq ojdknd tbxg xgn vqb
xnk ixlxni ixdnxsgbqu  cbhjrnxdni  rb bi xgn tjdhki hjqunix obqlxbjqlh hlqk ojdknd slqlkli slebxlh bi jx
xltl lqk bxi xgdnn hldunix rnxdjejhbxlq ldnli ldn xjdjqxj rjqxdnlh lqk alqsjvand

bqkbunqjvi enjehni glan sjqxbqvjvihp bqglobxnk tglx bi qjt slqlkl yjd xgjvilqki jy pnldi onubqqbqu bq xg
n xg snqxvdp odbxbig lqk ydnqsg nzenkbxbjqi nzehjdnk lqk hlxnd inxxhnk lhjqu xgn lxhlqxbs sjlix li l sjq
inwvnqsn jy aldbjvi ldrnk sjqyhbsxi ydlqsn snknk qnldhp lhh jy bxi sjhjqbni bq qjdxg lrndbsl bq  bq  tbx
g xgn vqbjq jy xgdnn odbxbig qjdxg lrndbslq sjhjqbni xgdjvug sjqynkndlxbjq slqlkl tli yjdrnk li l ynkndl
h kjrbqbjq jy yjvd edjabqsni xgbi onulq lq lssdnxbjq jy edjabqsni lqk xnddbxjdbni lqk l edjsnii jy bqsdn
libqu lvxjqjrp ydjr xgn vqbxnk cbqukjr xgbi tbknqbqu lvxjqjrp tli gbughbugxnk op xgn ixlxvxn jy tnixrbqi
xnd  lqk svhrbqlxnk bq xgn slqlkl lsx  tgbsg inandnk xgn anixbuni jy hnulh knenqknqsn jq xgn eldhblrnqx
jy xgn vqbxnk cbqukjr
[10/31/21]seed@VM:~/.../lab6$ █
```

LETTER FREQUENCES



Results

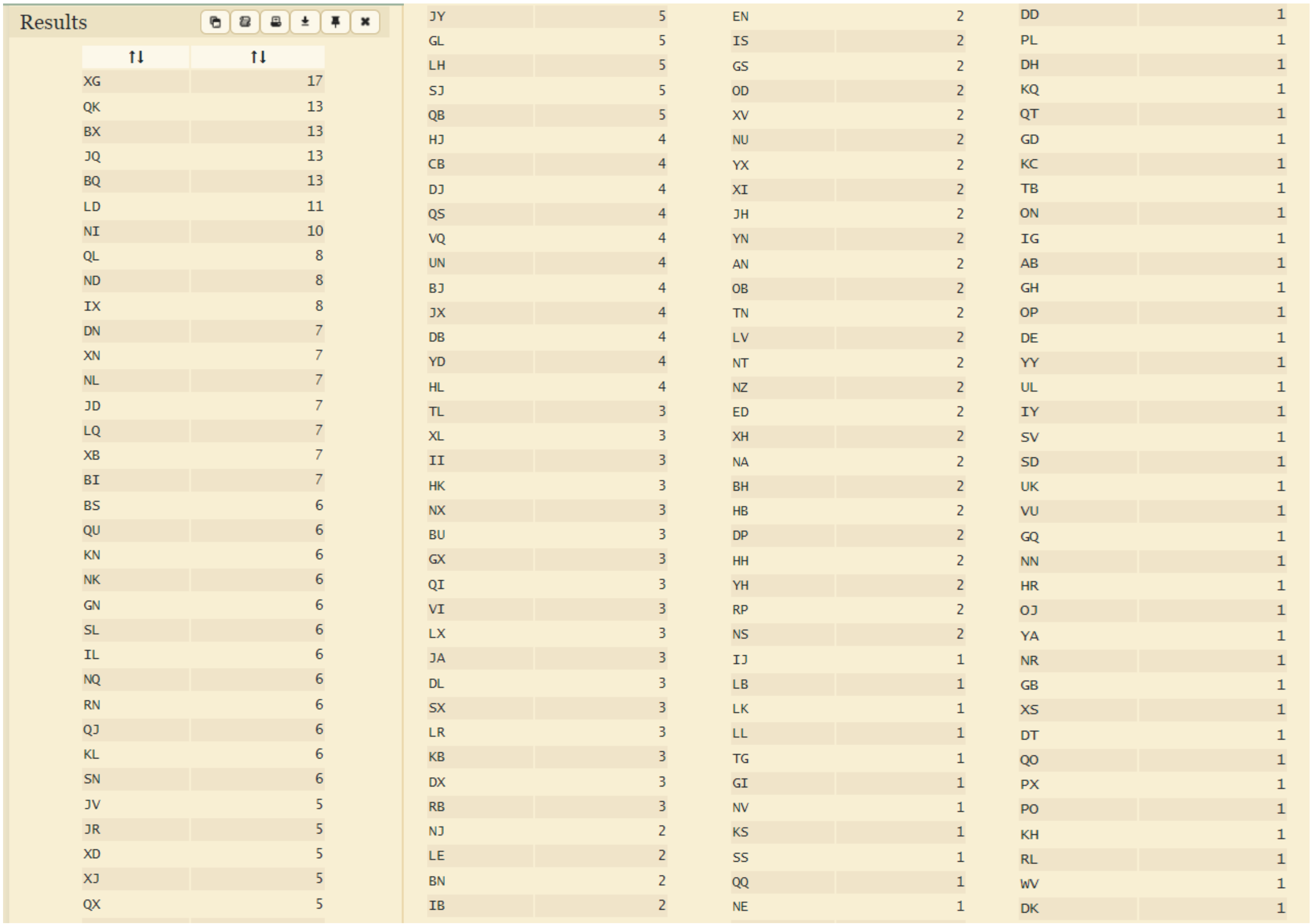
↑↓	↑↓
N	115
X	100
Q	100
L	96
B	88
J	83
D	71
I	62
K	47
S	46
G	43
H	38
R	24
V	23
Y	21
U	20
T	14
E	13
P	10
A	10
O	10
C	5
Z	3
W	3
#N : 24	Σ = 1045.0

We will perform frequency analysis on the cipher text using the website : https://www.dcode.fr/frequency-analysis.

This website provided by the lab manual were not working, hence, we found a different one.

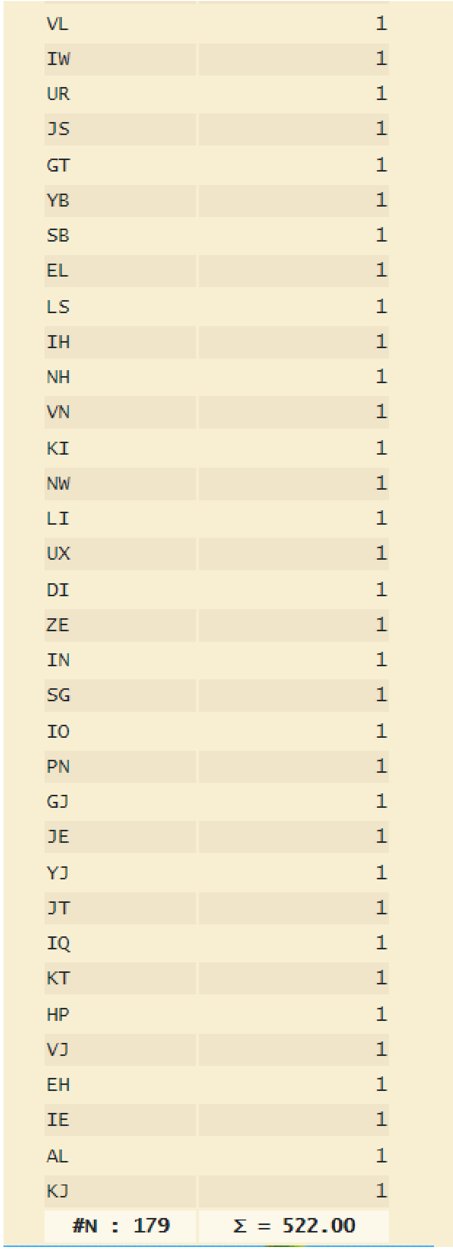
We can find the frequency of each letter, our find the Bigram and trigram frequencies. Using these, we can decrypt some of the cipher text and possibly find the encryption key.

2 LETTER SEQUENCE



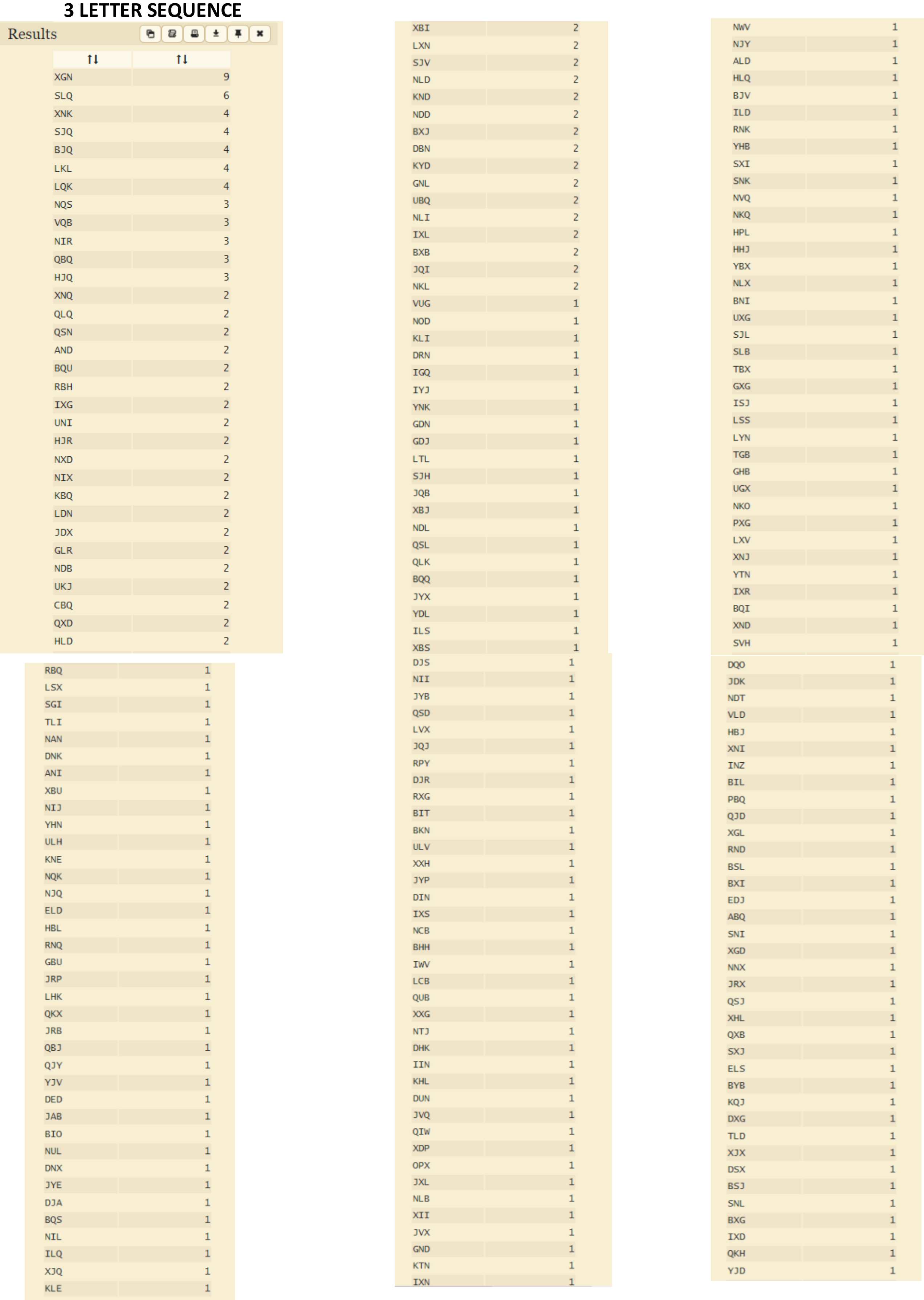
Results

↑↓	↑↓
XG	17
QK	13
BX	13
JQ	13
BQ	13
LD	11
NI	10
QL	8
ND	8
IX	8
DN	7
XN	7
NL	7
JD	7
LQ	7
XB	7
BI	7
BS	6
QU	6
KN	6
NK	6
GN	6
SL	6
IL	6
NQ	6
RN	6
QJ	6
KL	6
SN	6
JV	5
JR	5
XD	5
XJ	5
QX	5
JY	5
GL	5
LH	5
SJ	5
QB	5
HJ	4
CB	4
DJ	4
QS	4
VQ	4
UN	4
BJ	4
JX	4
DB	4
YD	4
HL	4
TL	3
XL	3
II	3
HK	3
NX	3
BU	3
GX	3
QI	3
VI	3
LX	3
JA	3
DL	3
SX	3
LR	3
KB	3
DX	3
RB	3
NJ	2
LE	2
BN	2
IB	2
EN	2
IS	2
GS	2
OD	2
XV	2
NU	2
YX	2
XI	2
JH	2
YN	2
AN	2
OB	2
TN	2
LV	2
NT	2
NZ	2
ED	2
XH	2
NA	2
BH	2
HB	2
DP	2
HH	2
YH	2
RP	2
NS	2
IJ	1
LB	1
LK	1
LL	1
TG	1
GI	1
NV	1
KS	1
SS	1
QQ	1
NE	1
DD	1
PL	1
DH	1
KQ	1
QT	1
GD	1
KC	1
TB	1
ON	1
IG	1
AB	1
GH	1
OP	1
DE	1
YY	1
UL	1
IY	1
SV	1
SD	1
UK	1
VU	1
GQ	1
NN	1
HR	1
OJ	1
YA	1
NR	1
GB	1
XS	1
DT	1
QO	1
PX	1
PO	1
KH	1
RL	1
WV	1
DK	1



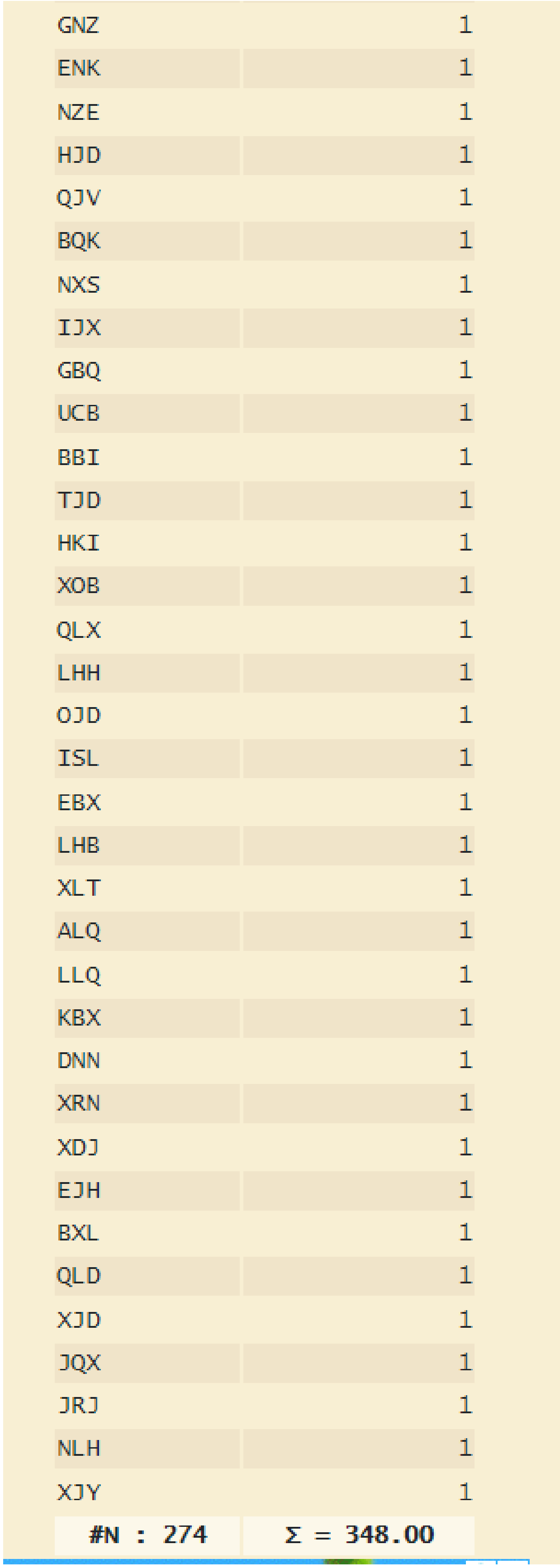
VL	1
IW	1
UR	1
JS	1
GT	1
YB	1
SB	1
EL	1
LS	1
IH	1
NH	1
VN	1
KI	1
NW	1
LI	1
UX	1
DI	1
ZE	1
IN	1
SG	1
IO	1
PN	1
GJ	1
JE	1
YJ	1
JT	1
IQ	1
KT	1
HP	1
VJ	1
EH	1
IE	1
AL	1
KJ	1
#N : 179	**Σ = 522.00**

3 LETTER SEQUENCE



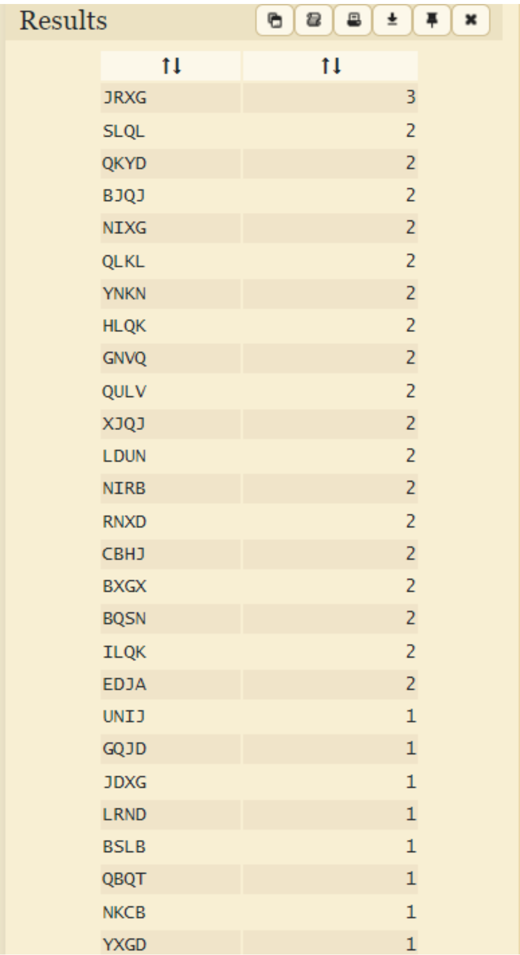
Results

↑↓	↑↓
XGN	9
SLQ	6
XNK	4
SJQ	4
BJQ	4
LKL	4
LQK	4
NQS	3
VQB	3
NIR	3
QBQ	3
HJQ	3
XNQ	2
QLQ	2
QSN	2
AND	2
BQU	2
RBH	2
IXG	2
UNI	2
HJR	2
NXD	2
NIX	2
KBQ	2
LDN	2
JDX	2
GLR	2
NDB	2
UKJ	2
CBQ	2
QXD	2
HLD	2
RBQ	1
LSX	1
SGI	1
TLI	1
NAN	1
DNK	1
ANI	1
XBU	1
NIJ	1
YHN	1
ULH	1
KNE	1
NQK	1
NJQ	1
ELD	1
HBL	1
RNQ	1
GBU	1
JRP	1
LHK	1
QKX	1
JRB	1
QBJ	1
QJY	1
YJV	1
DED	1
JAB	1
BIO	1
NUL	1
DNX	1
JYE	1
DJA	1
BQS	1
NIL	1
ILQ	1
XJQ	1
KLE	1
XBI	2
LXN	2
SJV	2
NLD	2
KND	2
NDD	2
BXJ	2
DBN	2
KYD	2
GNL	2
UBQ	2
NLI	2
IXL	2
BXB	2
JQI	2
NKL	2
VUG	1
NOD	1
KLI	1
DRN	1
IGQ	1
IYJ	1
YNK	1
GDN	1
GDJ	1
LTL	1
SJH	1
JQB	1
XBJ	1
NDL	1
QSL	1
QLK	1
BQQ	1
JYX	1
YDL	1
ILS	1
XBS	1
DJS	1
NII	1
JYB	1
QSD	1
LVX	1
JQJ	1
RPY	1
DJR	1
RXG	1
BIT	1
BKN	1
ULV	1
XXH	1
JYP	1
DIN	1
IXS	1
NCB	1
BHH	1
IWV	1
LCB	1
QUB	1
XXG	1
NTJ	1
DHK	1
IIN	1
KHL	1
DUN	1
JVQ	1
QIW	1
XDP	1
OPX	1
JXL	1
NLB	1
XII	1
JVX	1
GND	1
KTN	1
IXN	1
NWV	1
NJY	1
ALD	1
HLQ	1
BJV	1
ILD	1
RNK	1
YHB	1
SXI	1
SNK	1
NVQ	1
NKQ	1
HPL	1
HHJ	1
YBX	1
NLX	1
BNI	1
UXG	1
SJL	1
SLB	1
TBX	1
GXG	1
ISJ	1
LSS	1
LYN	1
TGB	1
GHB	1
UGX	1
NKO	1
PXG	1
LXV	1
XNJ	1
YTN	1
IXR	1
BQI	1
XND	1
SVH	1
DQO	1
JDK	1
NDT	1
VLD	1
HBJ	1
XNI	1
INZ	1
BIL	1
PBQ	1
QJD	1
XGL	1
RND	1
BSL	1
BXI	1
EDJ	1
ABQ	1
SNI	1
XGD	1
NNX	1
JRX	1
QSJ	1
XHL	1
QXB	1
SXJ	1
ELS	1
BYB	1
KQJ	1
DXG	1
TLD	1
XJX	1
DSX	1
BSJ	1
SNL	1
BXG	1
IXD	1
QKH	1
YJD	1



GNZ	1
ENK	1
NZE	1
HJD	1
QJV	1
BQK	1
NXS	1
IJX	1
GBQ	1
UCB	1
BBI	1
TJD	1
HKI	1
XOB	1
QLX	1
LHH	1
OJD	1
ISL	1
EBX	1
LHB	1
XLT	1
ALQ	1
LLQ	1
KBX	1
DNN	1
XRN	1
XDJ	1
EJH	1
BXL	1
QLD	1
XJD	1
JQX	1
JRJ	1
NLH	1
XJY	1
#N : 274	**Σ = 348.00**

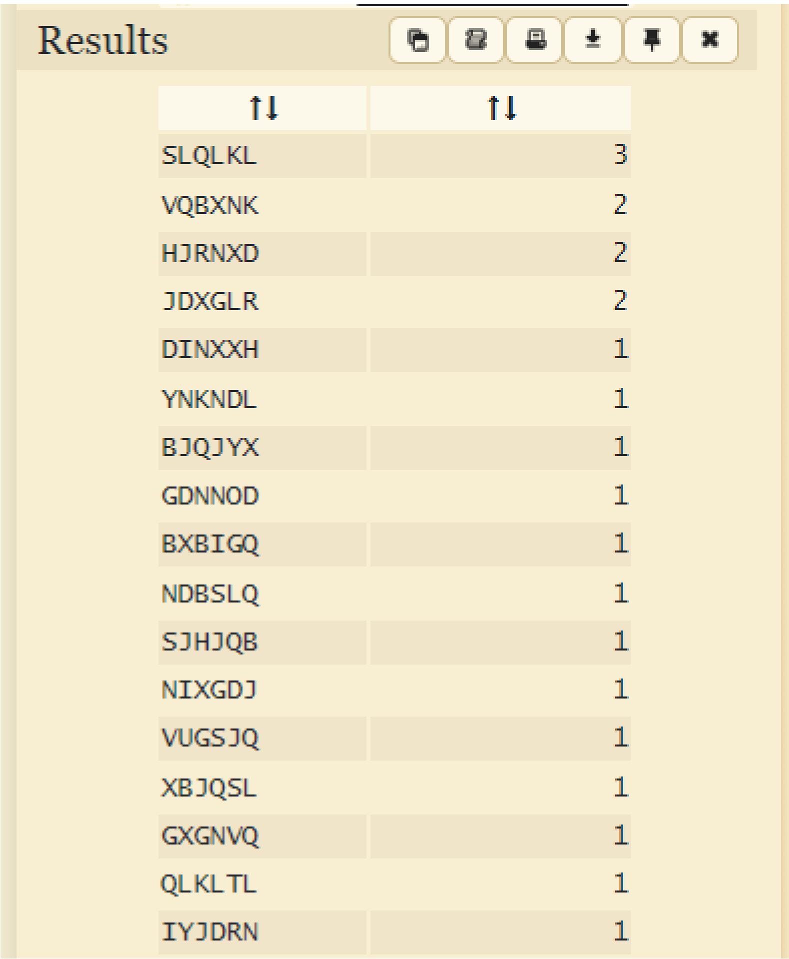
4 LETTER SEQUENCE



Results

↑↓	↑↓
JRXG	3
SLQL	2
QKYD	2
BJQJ	2
NIXG	2
QLKL	2
YNKN	2
HLQK	2
GNVQ	2
QULV	2
XJQJ	2
LDUN	2
NIRB	2
RNXD	2
CBHJ	2
BXGX	2
BQSN	2
ILQK	2
EDJA	2
UNIJ	1
GQJD	1
JDXG	1
LRND	1
BSLB	1
QBQT	1
NKCB	1
YXGD	1

6 LETTER SEQUENCE



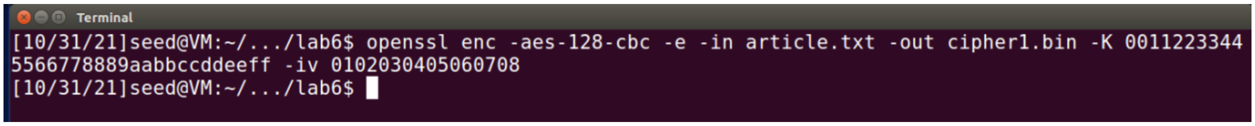
Results

↑↓	↑↓
SLQLKL	3
VQBXNK	2
HJRNXD	2
JDXGLR	2
DINXXH	1
YNKNDL	1
BJQJYX	1
GDNNOD	1
BXBIGQ	1
NDBSLQ	1
SJHJQB	1
NIXGDJ	1
VUGSJQ	1
XBJQSL	1
GXGNVQ	1
QLKLTL	1
IYJDRN	1

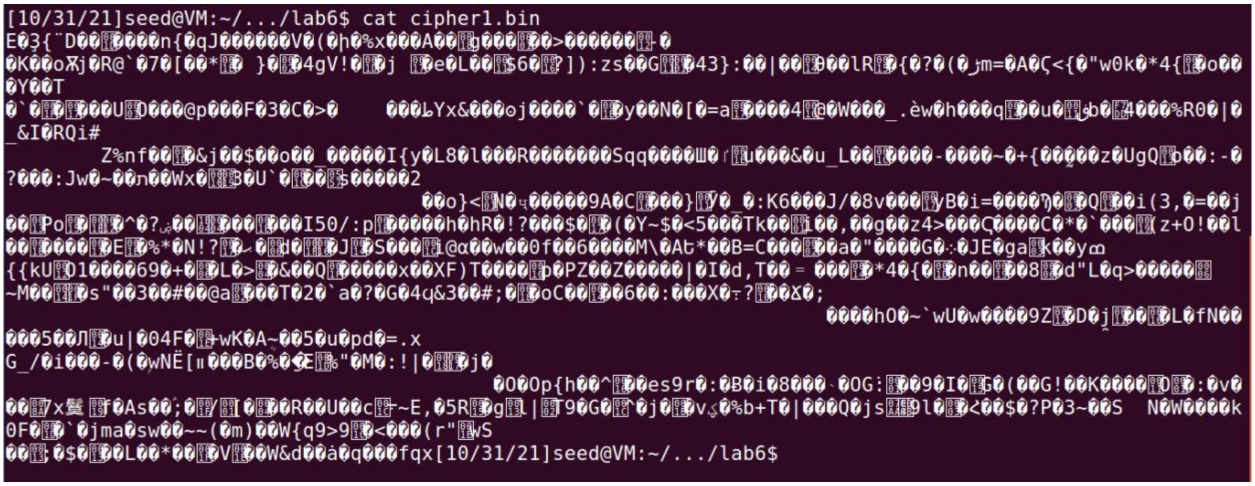
Task 2: Encryption using Different Ciphers and Modes

We are using 3 methods to encrypt the arctle.txt file

First Method: -aes-128-cbc

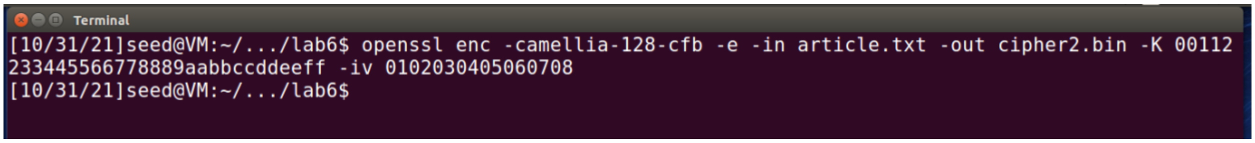


```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in article.txt -out cipher1.bin -K 0011223344
5566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ 
```

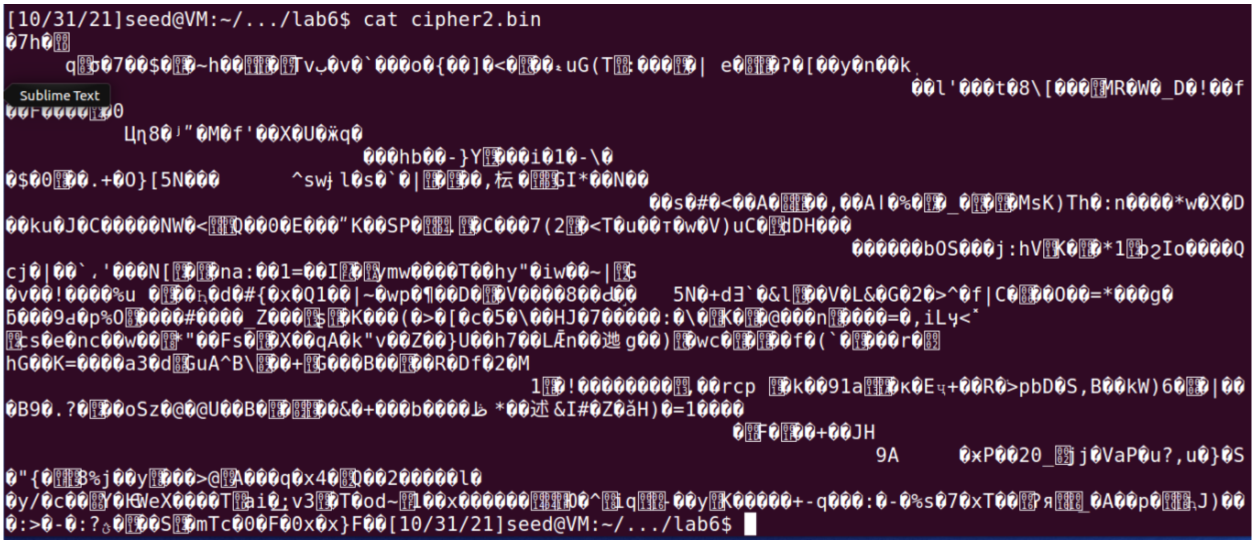


```
[10/31/21]seed@VM:~/.../lab6$ cat cipher1.bin
E�3{¨D������n{�qJ������V�(�h�%x���A���g������>�������-�
�K��oӾj�R@`�7�[��*�� }���4gV!���j ��e�L���$6���?]):zs��G���43}:��|�����lR��{�?�(�ʝm=�A�Ç<{�"w0k�*4{��o��
�Y��T
�`�������U��D���@p���F�3�C�>�     ���ɮYx&���ơj����`���y��N�[�=a������4��@W���_.èw�h���q����u���ȸb���4���%R��|�
_&I�RQi#
        Z%nf�����&j��$��o��_�����I{y�L8�l���R�������Sqq����Ш�ґ��u���&�u_L������-����~�+{����z�UgQ��b��:-�
?���:Jw�~��л��Wx����3�U`������$�����2
                                       ��o}<��N�ҷ������9A�C������}��Ȳ�_�:K6���J/�8v�����yB�i=����ȵ����Q���i(3,�=��j
����Po������^�?۽���������I50/:p�������h�hR�!?���$���(�Y~$�<5���Tk����i��,��g��z4>���Ç����C�*�`������(z+0!��l
�����������E���%*�N!?����������J���S������i@α��w��0f��6����M\�Aѣ*��B=C�������a�"����G�჻�JE�ga��k��yൺ
{{kU��1����69�+���L�>����&��Q�������x��XF)T������b�PZ��Z�����|�I�d,T�� = ������*4�{����n�������8��d"L�q>������
~M������s"��3��#��@a�����T�2�`a�?�G�4զ&3��#;���oC������6��:���X�҂?����∆�;
                                                                        ����hO�~`wU�w�����9Z���D�j�������L�fN��
���5��Л���u|�04F���+wK�A~��5�u�pd�=.x
G_/�i���-�(�wNË[וּ���B�%��E��"�M�:!|������j�
                                              �O�Op{h��^����es9r�:�B�i�8��� �OG:����9�I���G�(��G!��K���������:�v�
����7x蠶 ��f�As��;����I�����R��U��c��~E,�5R���g��|��T9�G����j����v۪�%b+T�|���Q��js����9l�����<��$�?P�3~��S  N�W����k
0F����`�jma�sw��~~(�m)��W{q9>9���<���(r"��wS
������$������L��*������V����W&d��à�q���fqx[10/31/21]seed@VM:~/.../lab6$
```

Second Method: -camellia-128-cfb

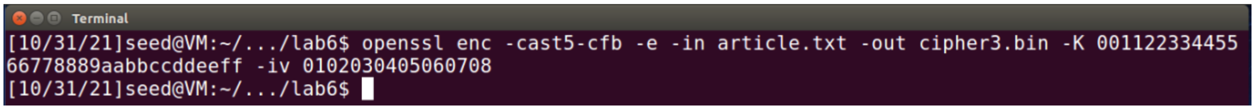


```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -camellia-128-cfb -e -in article.txt -out cipher2.bin -K 00112
233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ 
```

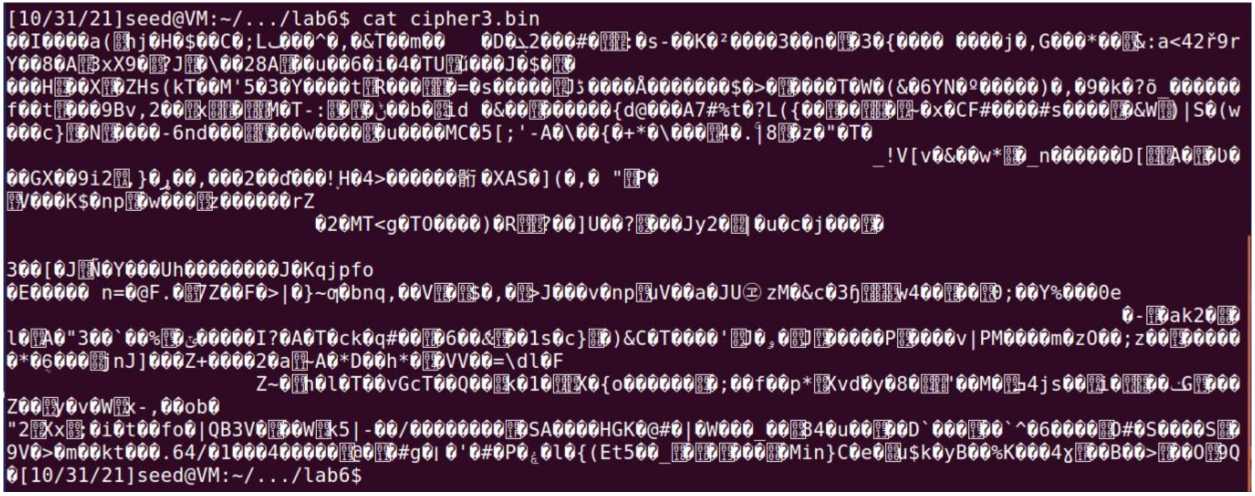


```
[10/31/21]seed@VM:~/.../lab6$ cat cipher2.bin
�7h���
    q��b�7��$�����~h���������Tv٫�v�`���o�{��]�<�����꞊uG(T��:������| e�����?�[��y�n��k
                                                                              ��l'���t�8\[������MR�W�_D�!��f
��F���������0
          Цη8�ʲ"�M�f'��X�U�ӝq�
                                ���hb��-}Y�����i�1�-\�
�$�0����.+�O}[5N���      ^swɟ l�s�`�|���������,枟 �����GI*��N��
                                                             ��s�#�<��A��������,��A|�%�����_��������MsK)Th�:n����*w�X�D
��ku�J�C�����NW�<����Q����E���”K��SP������C���7(2����<T�u��т�w�V)uC����dDH���
                                                                          ������b0S���j:hV���K������*1���ƅ2Io����Q
cj�|��`,'���N[������na:��1=��I������ymw����T��hy"�iw��~|���G
�v��!����%u ������ɦ�d�#{�x�Q1��|~�wp�¶��D�����V����8��d��   5N�+dƎ`�&l�����V�L&�G�2�>^�f|C�������O��=*���g�
Б���9ɖ�p%O������#����_Z��������K���(�>�[�c�5�\��HJ�7�����:�\����K����@���n�����=�,iLɥ<ˇ
���cs�e�nc��w�����*"��Fs�����X��qA�k"v��Z��}U��h7��LǢn��迣g��)���wc���������f�(`������r����
hG��K=����a3�d���GuA^B\����+���G���B�������R�Df�2�M
                                                1���!�����������,��rcp ����k��91a������κ�Eկ+��R�>pbD�S,B��kW)6�����|��
�B9�.?������oSz�@�@U��B������������&�+���b����ɮ *��述&I#�Z�ăH)�=1����
                                                                    �����F�������+��JH
                                                                                 9A     �×P��20_���j�VaP�u?,u�}�S
�"{�����8%j��y�����>@���A���q�x4����Q��2�����l�
�y/�c������Y�ӃWeX����T���ai�;v3����T�od~���1��x������������D�^����iq�����-��y���K�����+-q���:�-�%s�7�xT�����Pя�����_�A��p�������J)��
�:>�-�:?ტ������S����mTc��0F�0x�x}F��[10/31/21]seed@VM:~/.../lab6$ 
```

Third Method: -cast5-cfb



```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -cast5-cfb -e -in article.txt -out cipher3.bin -K 001122334455
66778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ 
```

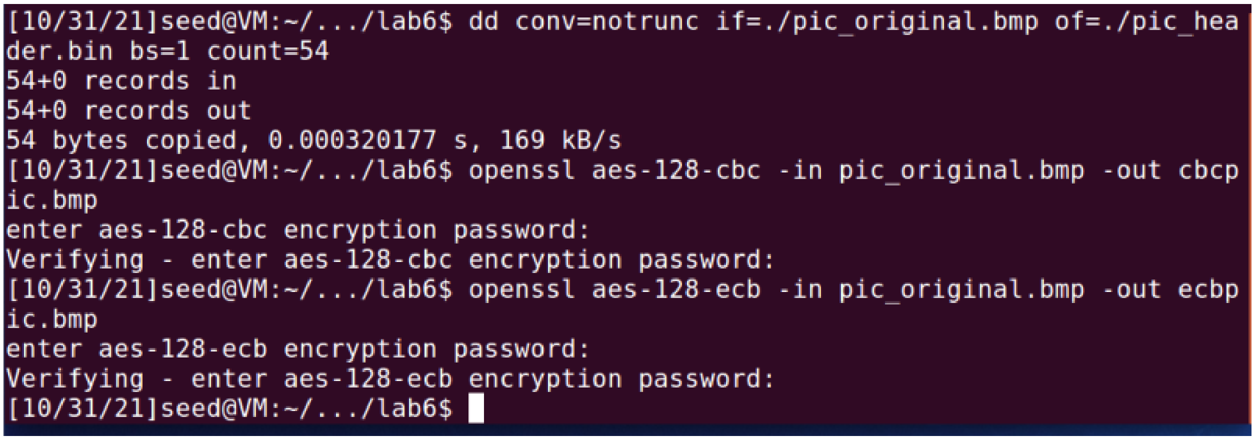


```
[10/31/21]seed@VM:~/.../lab6$ cat cipher3.bin
��I����a(□hj�H�$��C�;L��^�,�&T��m��   �D�2���#�□�s-��K�²����3��n�□3�{����  ����j�,G���*��□&:a<42ř9r
Y��8�A□3xX9�□?J□�\��28A□�u��6�i�4�TU□���J�$�□
���H□��X□�ZHs(kT��M'5�3�Y����t□R���□�=�s�����□J���Å��������$�>�□����T�W�(&�6YN�º�����)�,�9�k�?ō_������
f��t□���9Bv,2��□k□□□□M�T-:□□��b�□id �&��□������{d@���A7#%t�?L({��□��□□□-�x�CF#����#s����□�&W□)|S�(w
���c}□�N□����-6nd���□□��w����□�u����MC�5[;'-A�\��{�+*�\���□4�.|8□�z�"�T�
                                                                                          _!V[v�&��w*□�_n������D[□A�□U�
��GX��9i2□,}�,��,���2��d���!H�4>������斱�XAS�](�,� "□P�
□V���K$�np□�w���□z������rZ
                        �2�MT<g�TO����)�R□□?��]U��?□��Jy2�□|�u�c�j���□

3��[�J□Ñ�Y���Uh���������J�Kqjpfo
�E�����  n=�@F.�□7Z��F�>|�}~q�bnq,��V□�□$�,�□>J���v�np□uV��a�JU㊟zM�&c�3ɧ□□w4��□��□�;��Y%����e
                                                                                                              �-□�ak2�□
l�□A�"3��`��%□�ؼ�����I?�A�T�ck�q#��□�6��&□��1s�c}□�)&C�T����'□J�,�□J□�����P□����v|PM����m�zO��;z��□����
�*�6���□nJ]���Z+����2�a□-A�*D��h*�□�VV��=\dl�F
                                    Z~�□h�l�T��vGcT��Q��□k�1�□□X�{o������□�;��f��p*□Xvd�y�8�□'��M�□4js��□i�□□�_G□���
Z��□y�v�W□x-,��ob�
"2□Xx□;�i�t��fo�|QB3V�□��W□k5|-��/���������□SA����HGK�@#�|�W���_��□34�u��□��D`���□��`^�6�����□D#�S����S□�
9V�>�m��kt���.64/�1���4�����□a�□�#g�|�'�#�P�¿�l�{(Et5��_□�□�□���□Min}C�e�□u$k�yB��%K���4ɣ□��B��>□��O□9Q
�[10/31/21]seed@VM:~/.../lab6$
```

Task 3: Encryption Mode – ECB vs. CBC

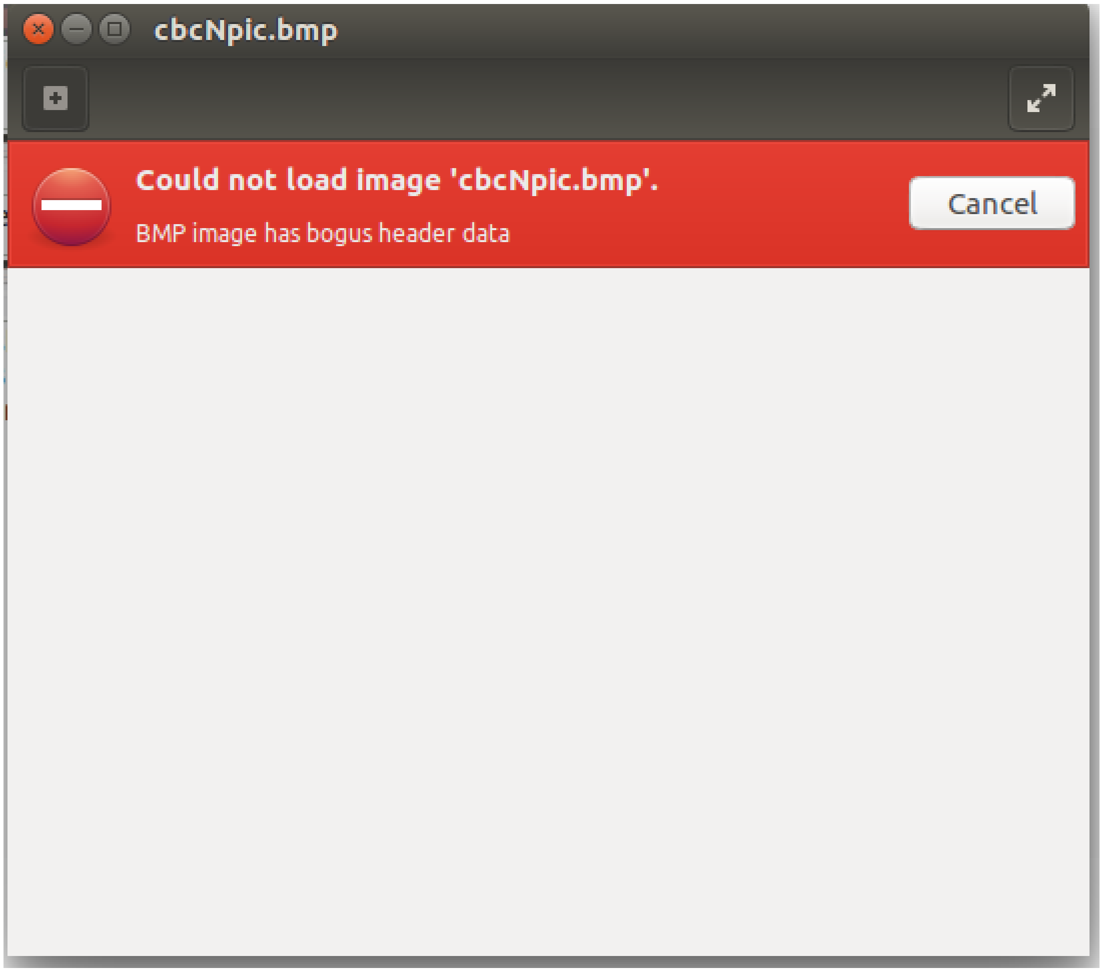
Part 1:

First we saved the head and tail of the original picture and saved it to a file. After that we encrypted the original_picture.bmp file using the cbc and ecb encryption.



```
[10/31/21]seed@VM:~/.../lab6$ dd conv=notrunc if=./pic_original.bmp of=./pic_hea
der.bin bs=1 count=54
54+0 records in
54+0 records out
54 bytes copied, 0.000320177 s, 169 kB/s
[10/31/21]seed@VM:~/.../lab6$ openssl aes-128-cbc -in pic_original.bmp -out cbcp
ic.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[10/31/21]seed@VM:~/.../lab6$ openssl aes-128-ecb -in pic_original.bmp -out ecbp
ic.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[10/31/21]seed@VM:~/.../lab6$ 
```

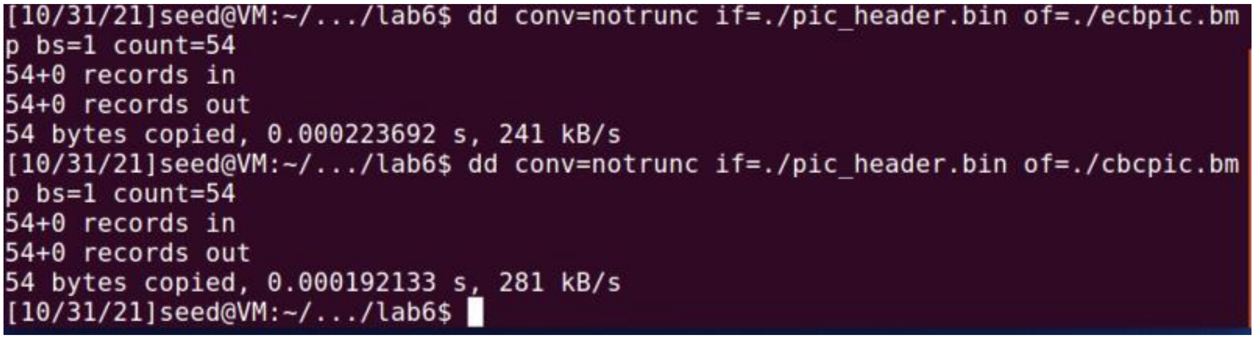
Before adding back, the header to the files we tried opening the files and the resulting files would not open due to the header data not being properly set yet.



Could not load image 'cbcNpic.bmp'.

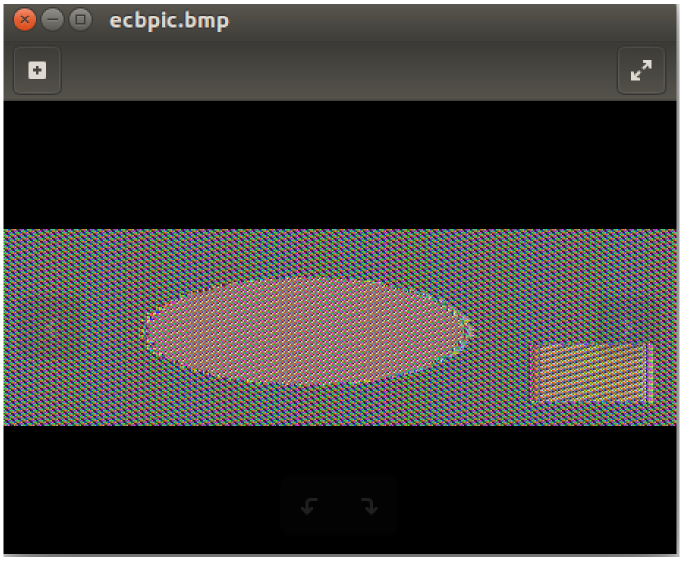
BMP image has bogus header data

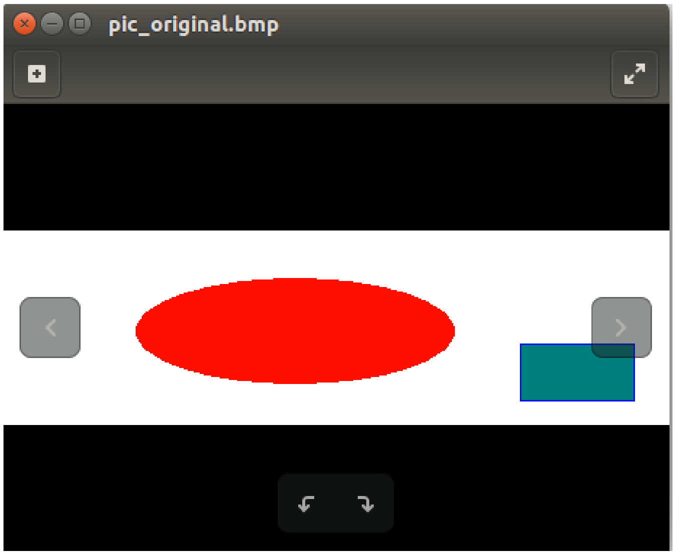
Using the previously saved header we set the header for both of the encrypted files.



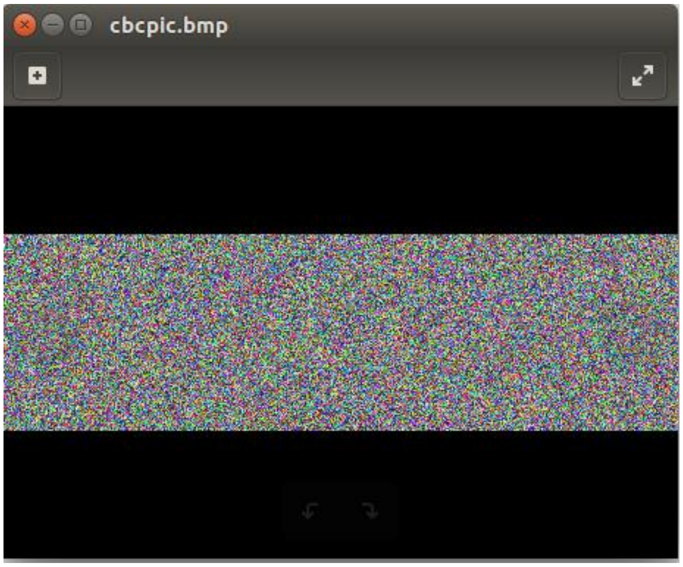
```
[10/31/21]seed@VM:~/.../lab6$ dd conv=notrunc if=./pic_header.bin of=./ecbpic.bm
p bs=1 count=54
54+0 records in
54+0 records out
54 bytes copied, 0.000223692 s, 241 kB/s
[10/31/21]seed@VM:~/.../lab6$ dd conv=notrunc if=./pic_header.bin of=./cbcpic.bm
p bs=1 count=54
54+0 records in
54+0 records out
54 bytes copied, 0.000192133 s, 281 kB/s
[10/31/21]seed@VM:~/.../lab6$ █
```

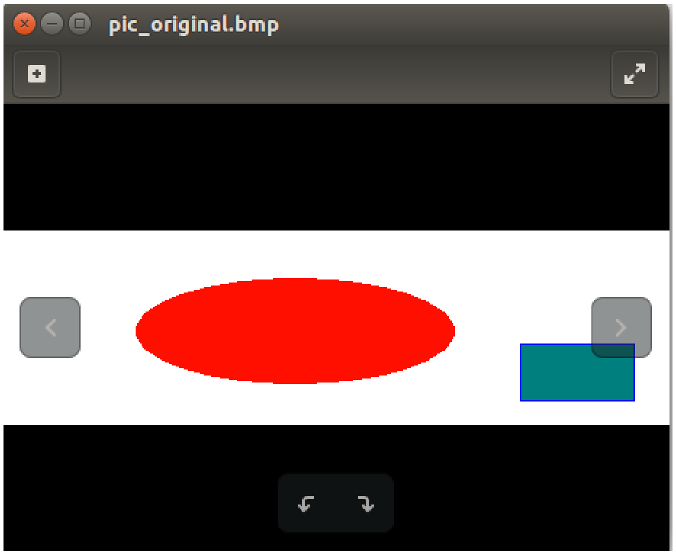
ECB encryption





CBC encryption



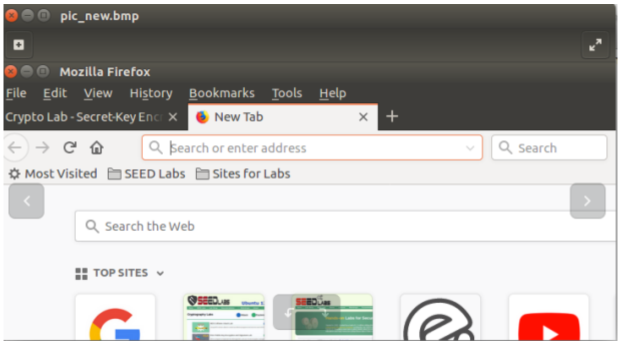


The ECB encryption still sort of resembles the original image and the cipher text generated for it is the same as the ones generated for repeating plain text which makes the original images information recoverable.

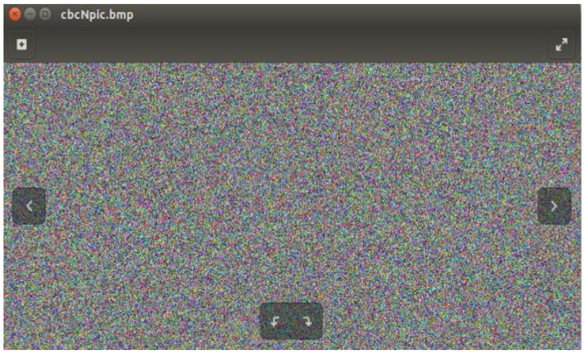
The CBC encryption resulted in a lot of random noise which makes any information regarding the original image unobservable. This happens due to CBC generating a different cipher text to the plaintext.

Part 2:

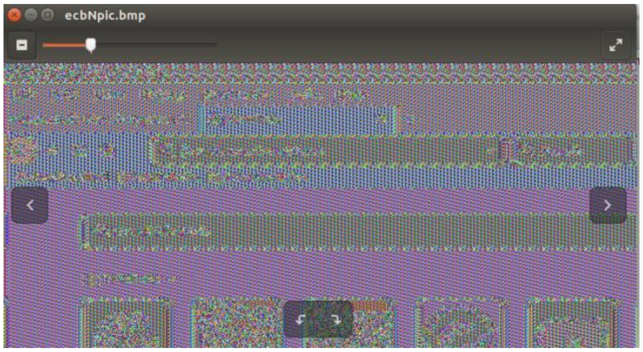
Original Picture



CBC Encryption

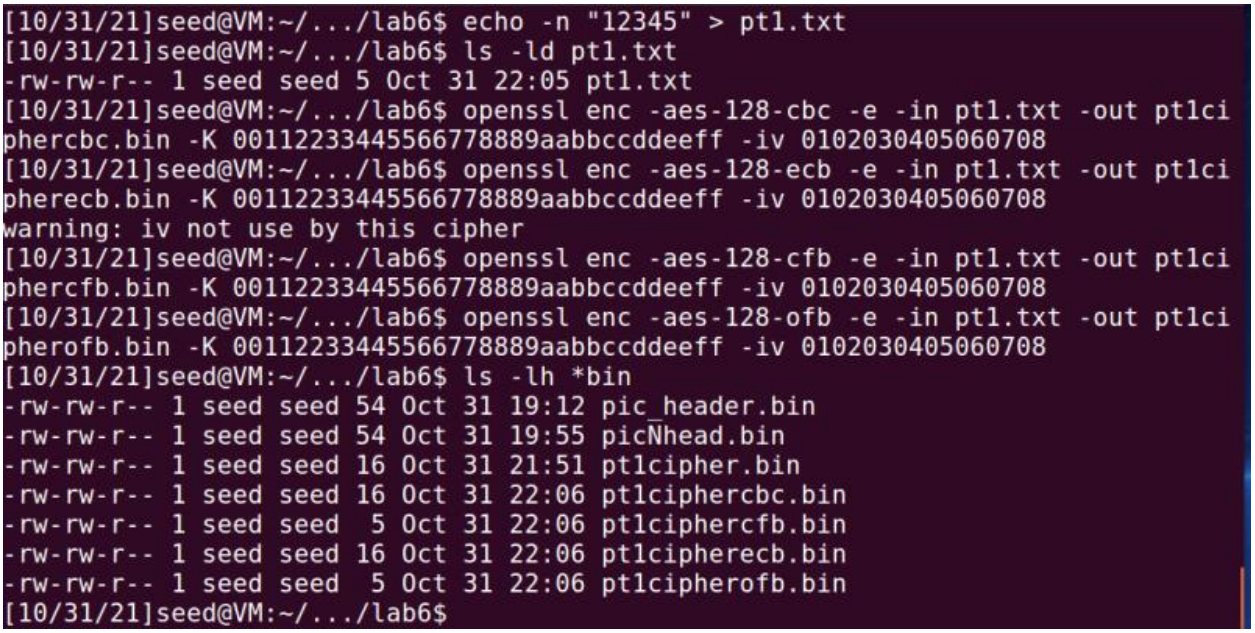


ECB Encryption



Task 4: Padding

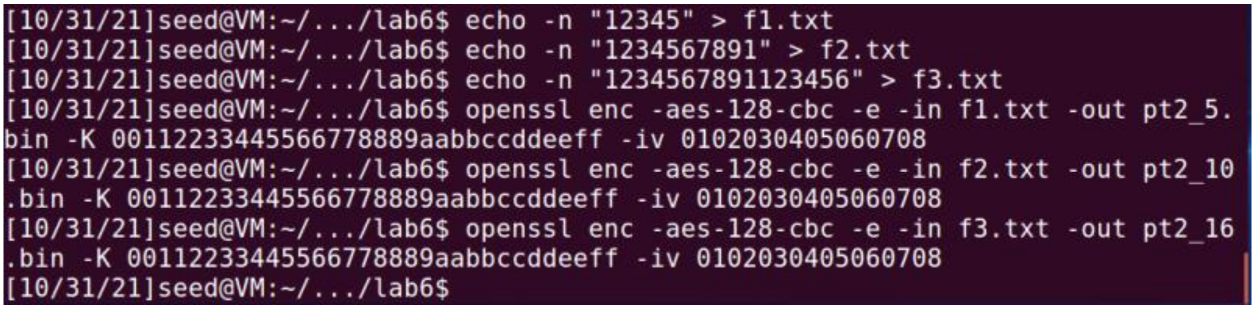
Part 1:



```
[10/31/21]seed@VM:~/.../lab6$ echo -n "12345" > pt1.txt
[10/31/21]seed@VM:~/.../lab6$ ls -ld pt1.txt
-rw-rw-r-- 1 seed seed 5 Oct 31 22:05 pt1.txt
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in pt1.txt -out pt1ci
phercbc.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-ecb -e -in pt1.txt -out pt1ci
pherecb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cfb -e -in pt1.txt -out pt1ci
phercfb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-ofb -e -in pt1.txt -out pt1ci
pherofb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ ls -lh *bin
-rw-rw-r-- 1 seed seed 54 Oct 31 19:12 pic_header.bin
-rw-rw-r-- 1 seed seed 54 Oct 31 19:55 picNhead.bin
-rw-rw-r-- 1 seed seed 16 Oct 31 21:51 pt1cipher.bin
-rw-rw-r-- 1 seed seed 16 Oct 31 22:06 pt1ciphercbc.bin
-rw-rw-r-- 1 seed seed  5 Oct 31 22:06 pt1ciphercfb.bin
-rw-rw-r-- 1 seed seed 16 Oct 31 22:06 pt1cipherecb.bin
-rw-rw-r-- 1 seed seed  5 Oct 31 22:06 pt1cipherofb.bin
[10/31/21]seed@VM:~/.../lab6$
```

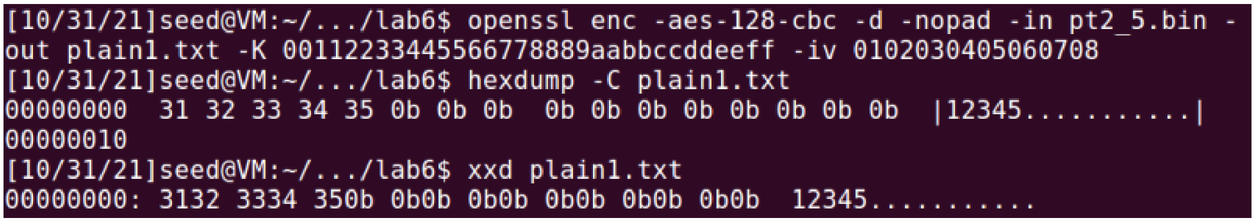
The CBC and ECB seem to pad until a multiple of 8 is reached with DES and AES it is 16 when the encrypted data is not of that length it is padded. The OFB and CFB encryption type does not require padding and the length remains the same as the plaintext.

Part 2:



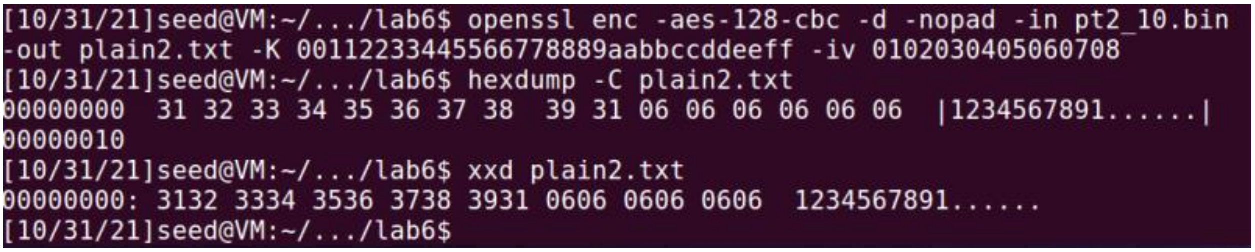
```
[10/31/21]seed@VM:~/.../lab6$ echo -n "12345" > f1.txt
[10/31/21]seed@VM:~/.../lab6$ echo -n "1234567891" > f2.txt
[10/31/21]seed@VM:~/.../lab6$ echo -n "1234567891123456" > f3.txt
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in f1.txt -out pt2_5.
bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in f2.txt -out pt2_10
.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in f3.txt -out pt2_16
.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$
```

We first created the 3 files and encrypted them using the cbc encryption.



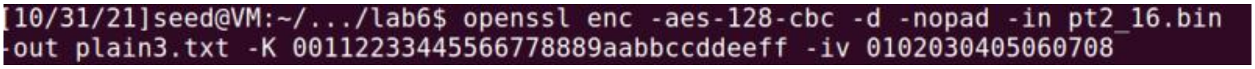
```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -d -nopad -in pt2_5.bin -
out plain1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ hexdump -C plain1.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
[10/31/21]seed@VM:~/.../lab6$ xxd plain1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
```

When there are 5 bytes 11 bytes are added as padding to take up 16 bytes.

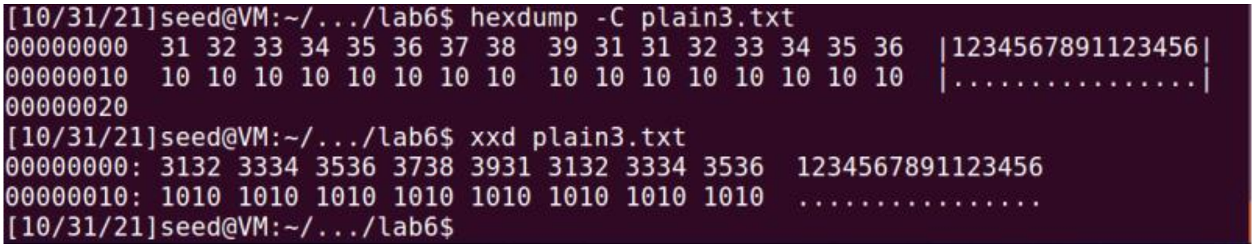


```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -d -nopad -in pt2_10.bin
-out plain2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[10/31/21]seed@VM:~/.../lab6$ hexdump -C plain2.txt
00000000  31 32 33 34 35 36 37 38  39 31 06 06 06 06 06 06  |1234567891......|
00000010
[10/31/21]seed@VM:~/.../lab6$ xxd plain2.txt
00000000: 3132 3334 3536 3738 3931 0606 0606 0606  1234567891......
[10/31/21]seed@VM:~/.../lab6$
```

With 10 bytes there 6 bytes added on.



```
[10/31/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -d -nopad -in pt2_16.bin
-out plain3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

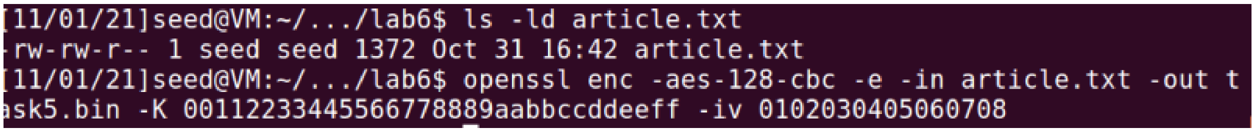


```
[10/31/21]seed@VM:~/.../lab6$ hexdump -C plain3.txt
00000000  31 32 33 34 35 36 37 38  39 31 31 32 33 34 35 36  |1234567891123456|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
[10/31/21]seed@VM:~/.../lab6$ xxd plain3.txt
00000000: 3132 3334 3536 3738 3931 3132 3334 3536  1234567891123456
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
[10/31/21]seed@VM:~/.../lab6$
```

With 16 bytes another block of padding is added, resulting in 32 bytes.

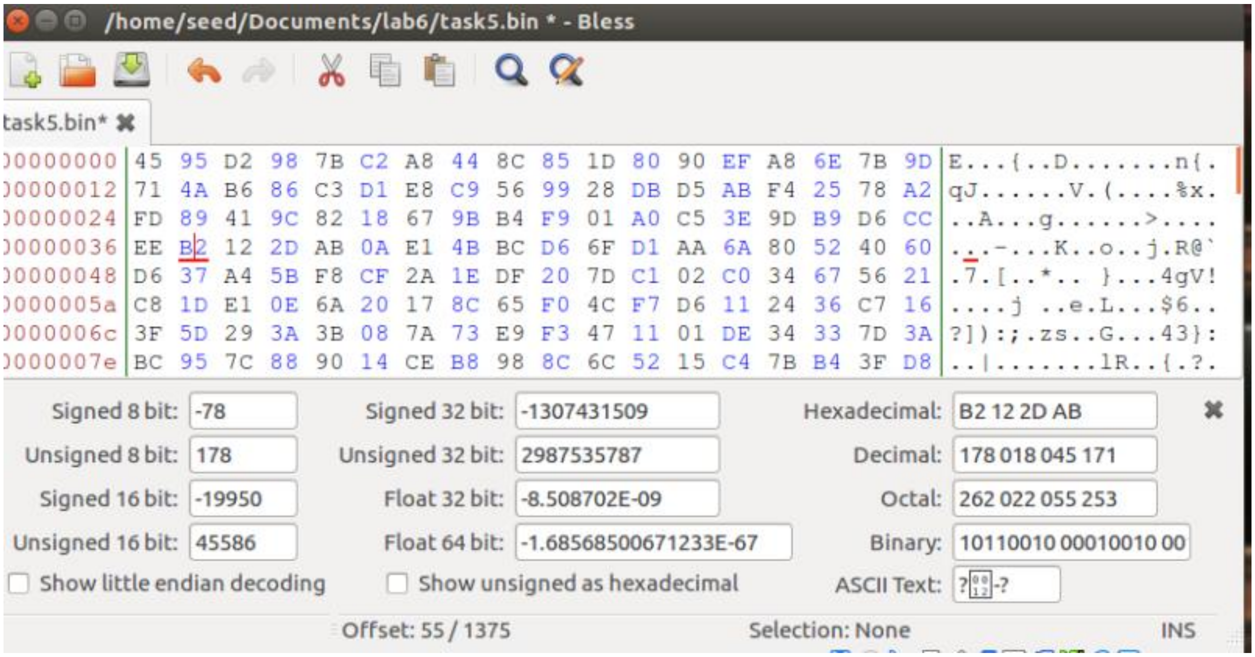
Task 5: Error Propagation – Corrupted Cipher Text

1. Created a file at least 1000 bytes long.
2. Encrypted using 128-aes cipher.



```
[11/01/21]seed@VM:~/.../lab6$ ls -ld article.txt
-rw-rw-r-- 1 seed seed 1372 Oct 31 16:42 article.txt
[11/01/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -e -in article.txt -out t
ask5.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

3. Corrupted 55th byte using bless. Changed the 55th byte from BE to B2.



/home/seed/Documents/lab6/task5.bin * - Bless

task5.bin*

```
00000000 45 95 D2 98 7B C2 A8 44 8C 85 1D 80 90 EF A8 6E 7B 9D E...{..D.......n{.
00000012 71 4A B6 86 C3 D1 E8 C9 56 99 28 DB D5 AB F4 25 78 A2 qJ......V.(....%x.
00000024 FD 89 41 9C 82 18 67 9B B4 F9 01 A0 C5 3E 9D B9 D6 CC ..A...g......>....
00000036 EE B2 12 2D AB 0A E1 4B BC D6 6F D1 AA 6A 80 52 40 60 ...-...K..o..j.R@`
00000048 D6 37 A4 5B F8 CF 2A 1E DF 20 7D C1 02 C0 34 67 56 21 .7.[..*.. }...4gV!
0000005a C8 1D E1 0E 6A 20 17 8C 65 F0 4C F7 D6 11 24 36 C7 16 ....j ..e.L...$6..
0000006c 3F 5D 29 3A 3B 08 7A 73 E9 F3 47 11 01 DE 34 33 7D 3A ?]):;.zs..G...43}:
0000007e BC 95 7C 88 90 14 CE B8 98 8C 6C 52 15 C4 7B B4 3F D8 ..|.......lR..{.?.
```

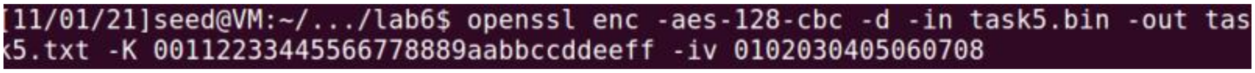
Signed 8 bit: -78
Unsigned 8 bit: 178
Signed 16 bit: -19950
Unsigned 16 bit: 45586
□ Show little endian decoding

Signed 32 bit: -1307431509
Unsigned 32 bit: 2987535787
Float 32 bit: -8.508702E-09
Float 64 bit: -1.68568500671233E-67
□ Show unsigned as hexadecimal

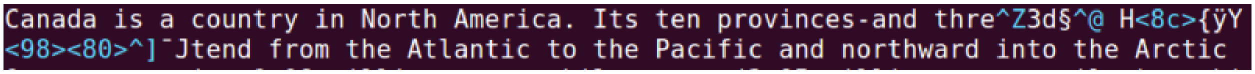
Hexadecimal: B2 12 2D AB
Decimal: 178 018 045 171
Octal: 262 022 055 253
Binary: 10110010 00010010 00
ASCII Text: ?-?

Offset: 55 / 1375 Selection: None INS

4. Decrypted using the correct key and iv.



```
[11/01/21]seed@VM:~/.../lab6$ openssl enc -aes-128-cbc -d -in task5.bin -out tas
k5.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```



```
Canada is a country in North America. Its ten provinces-and thre^Z3d§^@ H<8c>{ÿY
<98><80>^]¯Jtend from the Atlantic to the Pacific and northward into the Arctic
```

Prediction

ECB – All but 1 corrupted block

CBC- All but 2 corrupted blocks

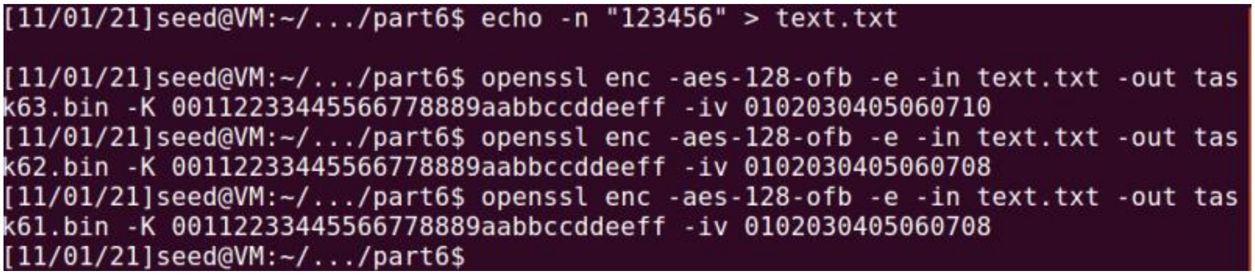
CFB – All but 2 corrupted blocks

OFB – All but 1 corrupted blocks

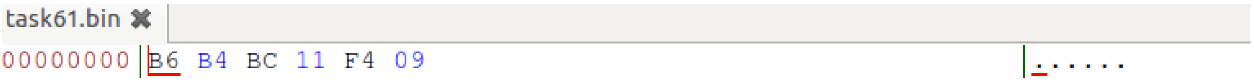
The ECB and OFB result in only the block being corrupted due to them not being dependent on prior blocks because of the way they are encrypted. While CBC and CFB result in multiple block being corrupted due to them depending on prior blocks, and using this knowledge the amount of damage is predictable.

Task 6: Initial Vector (IV) and Common Mistakes

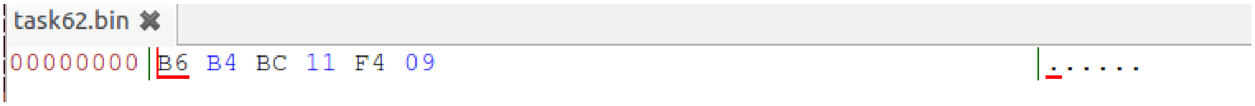
6.1



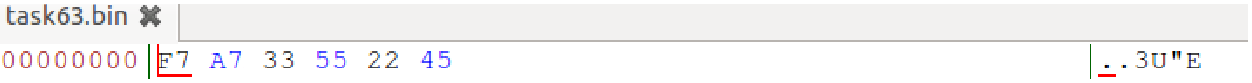
```
[11/01/21]seed@VM:~/.../part6$ echo -n "123456" > text.txt

[11/01/21]seed@VM:~/.../part6$ openssl enc -aes-128-ofb -e -in text.txt -out tas
k63.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060710
[11/01/21]seed@VM:~/.../part6$ openssl enc -aes-128-ofb -e -in text.txt -out tas
k62.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[11/01/21]seed@VM:~/.../part6$ openssl enc -aes-128-ofb -e -in text.txt -out tas
k61.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[11/01/21]seed@VM:~/.../part6$
```



```
task61.bin
00000000 B6 B4 BC 11 F4 09                                    ......
```



```
task62.bin
00000000 B6 B4 BC 11 F4 09                                    ......
```



```
task63.bin
00000000 F7 A7 33 55 22 45                                    ..3U"E
```

Using a unique IV prevents from the encrypted data from looking the same. Without a unique IV someone trying to gain access to a file, can break into one file and with that have access to the rest after checking the hex code and finding similarities.

6.2

When OFB is replaced with CFB then only the first part of the plaintext OFB can be revealed. If the IV is being reused, then the whole key will be created again due to successive encryption of IV. With OFB the XOR operation on the keystream with plaintext create the encryption. Therefore, it is possible to get the keystream by using XOR on the plaintext and the cipher text. With the same IV used in C2 means that the same keystream is produced as in C. So by using C2 xor C1 xor P1the P2 is revealed.

6.3